Amendments to the Drawings:

The attached replacement sheets of drawings includes changes to FIG. 1 to add "a and b" to "24" and FIG. 3 to replace the term "BACKGROUND" with the term "PRIOR" as requested by the Examiner. These sheets replace the original sheets having FIG. 1 and FIG. 3 only.

REMARKS

Applicant respectfully requests reconsideration of the present application in view of the amendments set forth above and the below remarks.

Claims 1-13 are pending in the application and are rejected.

The Drawing Objections

Figure is 2 is objected to as containing reference numbers not recited in the specification. The specification is amended as set forth above to overcome the objection.

Figure 3 is amended to replace the legend term "BACKGROUND" with the term "PRIOR" as requested by the Examiner.

Figure 1 is amended to show reference numbers 24a and 24b instead of 24 in two locations. The specification is amended for consistency with this change.

The Specification Objections

Line 9, paragraph [0036], line 9 is amended to remedy the typographical error pointed out by the Examiner.

The Prior Art Rejections

The Examiner rejects claims 1-13 under 35 U.S.C. §103 over U.S. Patent No. 5,720,026 to Uemura et al in view of Levy et al, *Incremental Recovery in Main Memory Database Systems*, and U.S. Patent No. 6,038,665 to Bolt et al.

The Examiner asserts that Uemura substantially teaches the claimed invention. This is clearly not correct and is clearly not supportable for the reasons set forth below in detail.

Claim 1 requires a method for *incrementally backing up data* from a logically represented volume on disk media, comprising:

identifying *tracks* of the logically represented volume *that have changed since a last incremental backup operation* by reading fresh data indications, (i) wherein each of the fresh data indications corresponds to a *track* of the logically represented volume and (ii) wherein a given fresh data indication is indicative of whether its corresponding track has been changed since a last incremental backup operation;

identifying *files* for incremental backup, the identified files comprising ***changed*** *and* ***unchanged*** *blocks* saved on a *track* deemed changed since a last incremental backup operation; and

incrementally backing up the identified files from the disk media to sequential storage media through a high speed connection.

Claim 1 requires, among other things:

- identifying tracks that have changed;

- identifying files for incremental backup, the identified files comprising changed and unchanged blocks saved on a track deemed changed since a last incremental backup operation; and

- incrementally backing up the identified files.

In contrast, Uemura teaches incremental backup of *changed blocks* of data in a storage unit. More particularly, Uemura discloses an incremental backup system having a storage unit that is accessed in block units for storing data to be backed up. Uemura teaches a "difference management mechanism for managing difference data in disk ***blocks***." (col. 4, lines 44-45, emphasis added). As shown in Figure 6 of Uemura, for each *block* the generation in which it is backed up is noted. Uemura simply does not contemplate any of identifying tracks or identifying files for incremental backup as required by claim 1.

The Examiner asserts that Uemura teaches the claimed step of "identifying tracks" by relying upon col. 8, lines 42-61, which is set forth below:

> To execute the incremental backup, if the system manager wants, he or she can also ask the control command 207 about the tape capacity required for the backup from the number-of-update-blocks table 500 before tape is loaded.

> At the termination of business on Tuesday with the incremental backup generation as 1, the incremental backup in generation 1 is executed by the control command 207. When the incremental backup with the generation scope as 1 is requested of the control command 207, the control command 207 informs that the incremental backup scope is 1 through the ioctl interface of the logical volume management mechanism 206. After this, the control command 207 simply executes read from the pseudo interface and simply writes into the tape drive. When a read request comes through the pseudo interface, the difference management mechanism 203 references the current incremental backup generation and the scope, searches the difference map information 600 for *the corresponding **block***, and creates sequential data as shown in FIG. 9. (emphasis added)

Applicant points out that the "difference map" provides a "change history for each block." (col. 5, lines 8-9). A review of Uemura clear shows that the disclosure is limited to determining changed blocks to perform an incremental backup.

The passage above cited by the Examiner does not teach or suggest the claimed step of identifying tracks. Claim 1 includes "identifying tracks" and identifying files," "the identified files comprising changed and unchanged blocks," and "incrementally backing up the identified files." Uemura is clearly limited to changes in blocks of data.

To teach the claim step of "identifying files for incremental backup, the files comprising changed and unchanged blocks," the Examiner points to col. 4, lines 34-64 and col. 10, lines 4-67, which are set forth below:

> The full backup is to back up all of data to be backed up in the disk units 111 and 112 and the incremental backup is to back up **only the data updated since the most recent backup**, which will be hereinafter referred to as **difference data**.
>
> FIG. 2 is a block diagram showing an example of the software configuration of the computer system to which the invention is applied.
>
> In FIG. 2, numeral 201 is an application program which references/updates disk data and numeral 202 is an operating system (the embodiment of the invention uses UNIX). Numeral 203 is a difference management mechanism for managing *difference data in disk **blocks***. It is built in the system as a pseudo device driver under the control of the operating system. Numeral 204 is a latest update generation management mechanism for managing backup generation numbers for *each **block*** in the disk units 111 and 112. Numeral 205 is a number-of-update-blocks management mechanism for managing the *number of update blocks* in the disk units 111 and 112 for each backup generation. Numeral 206 is a logical

volume management mechanism for managing virtual logical disk units made up of physical disk units like the disk units 111 and 112. The difference management mechanism 203, the latest update generation management mechanism 204, the number-of-update-blocks management mechanism 205 for each generation, and the logical volume management mechanism 206 are in the same position when viewed from the system. Numeral 207 is a control command used by the system manager for controlling the difference management mechanism 203. The control command can be used to execute full backup and incremental backup. (emphasis added)

...

The user can use a sequential read/write command, such as a dd command in the UNIX system, via the pseudo device driver interface to save difference **block** management data and difference data on sequential access storage media like tape without considering the actually updated blocks. To restore incremental backup data, the incremental backup data saved via the pseudo device driver interface can be written into the pseudo device driver interface in sequence for the backup volume where data to the generations preceding the incremental backup is already restored.

Further, the pseudo driver interface is provided with a function of adding check sum data for *each block data* and the check sum data for *each block data* is stored, thereby improving reliability of the incremental backup data.

If data is compressed and stored on the save volume at the incremental backup time and the compressed data is decompressed and restored at the incremental backup data restoring time, the data amount of the incremental backup data can be reduced and the effective use of the backup media can be made.

Embodiment 2:

FIG. 10 is a block diagram showing the hardware configuration of an incremental backup system in a second embodiment of the invention. Components identical with or similar to those previously described with reference to FIG. 1 are denoted by the same reference numerals in FIG. 10 and will not be discussed again.

In FIG. 10, numeral 113 is an SCSI adapter as a controller connected to an input/output bus 105; a difference management mechanism 203 is provided as an adapter function by firmware. Numeral 114 is an SCSI bus deriving from the SCSI adapter 113 and numerals 902 and 906 are disk packs connected to the SCSI bus 114. In addition to the two disk packs 902 and 906, three disk packs 903-905 (not shown) are also connected to the SCSI bus 114. Numeral 901 is a tape drive in which backup of data in the disk packs 902-906 is stored. The disk packs 902-906 are to be backed up by an incremental backup function provided by the SCSI adapter 113.

> From an operating system, it appears that the five disk packs 902-906 each containing an SCSI controller at target level are connected to the SCSI adapter 113. The SCSI specifications allow a maximum of eight target controllers on a single bus, three of which (ID7, ID6, and ID5) are used by a map storage disk of a disk unit 900 with an incremental backup function having a front panel shown in FIG. 11, the tape drive, and the SCSI adapter 113, and cannot be used by the operating system. In FIG. 11, tape drive 901 and disk packs 902-906 are mounted. The tape drive 901 and disk packs 902-906 in FIG. 11 are the same as those in FIG. 10. Numeral 907 is a full backup switch, numeral 908 is a full restore switch, numeral 909 is an incremental backup switch, numeral 910 is an incremental backup confirmation switch, numeral 911 is a difference restore switch, numeral 912 is a difference restore confirmation switch, and numeral 913 is a ten-key pad.
>
> An application program and the operating system in the second embodiment of the invention are the same as those in the first embodiment of the invention. Since a device driver of the unit does not time out, the application program using the disk packs 902-906 in the unit automatically stops during execution of backup or restoring by the unit. Therefore, the system manager can execute backup by the unit at any time.

The above passages from Uemura do not teach or suggest the claimed step of "identifying files for incremental backup, the files comprising changed and unchanged blocks."

The Examiner relies upon Levy to teach "fresh/stale markings."

Levy is directed to "recovery activities, like checkpointing and restart." Levy discloses a technique for performing recovery incrementally and in parallel with transactions to a database. This has no relevance to the claimed incremental backup invention. Levy teaches that after a database crash, a stale/fresh marker indicates which pages are stale and need to be updated. A log record can be used to determine which database updates are needed. The "marking enables resuming transaction processing immediately after a crash, while preserving the consistency of the database."

The Examiner states that the technique taught by Levy "would optimize the recovery speed of post-crash transactions." Applicant does not dispute that Levy may teach techniques to optimize recovery speeds. Applicant submits that this has no relevance to the claimed incremental backup invention.

The Examiner then acknowledges that "Uemura et al. and Levy et al. do not specifically disclose that the identified files comprise unchanged blocks saved on track." and points to Bolt at col. 9, lines 22-40 and col. 13, lines 4-42, which are set forth below:

> If, however, the *digital signature* MD5 of the **block** having as its first byte the byte.sub.k under test is determined to be equal to one of the digital signatures MD5.sup.old in the ordered list at decision diamond 75, the logic returns "resynchronized" and moves to block 76. In other words, a *positive test* at decision diamond 75 indicates that the logic has found *an old, unchanged* ***block*** that *previously has been backed up*, and, hence, that the *logic is resynchronized* with the comparison value listing.
>
> At block 76, the *changed* ***block***(s) (also referred to herein as "transmission blocks") are moved to a "next chunk" file. Additionally, at block 76 the comparison value listing is updated to include the first two characters and digital signatures of the changed block(s), for use as the first and second comparison values, respectively, during the test of the blocks during the next back up cycle. Moving to decision diamond 78, it is determined whether the chunk file is full. In the presently preferred embodiment, the chunk file is full when its size is five megabytes (5 MB).
>
> ...
>
> As disclosed below, in a portable computer environment, the present invention copies file blocks when the host portable computer is not connected to a network, and when the present invention senses a network connection, the copied blocks are transmitted via the network to a storage facility.
>
> Commencing at block 180, when the host portable computer is energized, the logic determines *whether all files on the user's computer are to be considered for back up, or whether only a user-defined set of files is to be considered.* If the user has defined a set of files (by, e.g., directory) for back up, using, for example, file inclusion/exclusion lists with wildcards, this set is received at block 182. From block 182, or from decision diamond 180 if no user-defined set is received and all files (or a default set of files) are to be candidates for back up, the logic moves to block 184 to prepare the blocks in, e.g., chunks as described above, for back up preferably in accordance with the logic set forth in FIG. 3 above. As mentioned above, the back up process is undertaken in the "background", transparently to the user as the user employs the host computer for other tasks such as, e.g., word processing or presentation slide generation.
>
> Decision diamond 186 represents that the logic of the present invention monitors for whether a predetermined storage space limit has been reached on the host portable computer. As recognized by the present invention, a portable computer such as a laptop computer that is energized but disconnected from a network might

> encounter storage space limitations due to the generation of duplicate blocks for back up, and the step at decision diamond 186 is to ensure that the back up process set forth herein, which is intended to be a "background" process, does not fill up the user's storage. In the preferred network embodiment, no more than 5% of the host computer's storage capacity is used for back up storage. Alternatively, the user can define the predetermined storage space limit. If a storage space limit has been reached, the logic moves to block 188 to suspend the back up process until, e.g., subsequent transmission of the blocks to be backed up followed by deletion of the back up copies on the host computer frees additional storage space.

The Examiner appears to place particular emphasis on the ability of Bolt to allow "for *user-defined* sets in defining blocks of data for backup." However, such emphasis is misplaced. While the meaning may be unclear, two interpretations are possible, neither of which provides any teaching or suggestion of changed and unchanged blocks. One possible interpretation is that the user selects files for backup and blocks are examined for changes as set forth in Figure 3.

> Commencing at block 180, when the host portable computer is energized, the logic determines whether all files on the user's computer are to be considered for back up, or whether only a user-defined set of files is to be considered. If the user has defined a set of files (by, e.g., directory) for back up, using, for example, file inclusion/exclusion lists with wildcards, this set is received at block 182. From block 182, or from decision diamond 180 if no user-defined set is received and all files (or a default set of files) are to be *candidates for back up*, the logic moves to block 184 to prepare the *blocks* in, e.g., chunks as described above, for back up preferably in accordance with the *logic set forth in **FIG. 3*** above. (emphasis added)

That is, changed blocks are identified by examining the digital signature as in Figure 3 of Uemura and marked for backup by placing them in the chunk file. In this case, *no unchanged blocks are backed up*. Applicant points out that Bolt, at col. 8, lines 40-47, states (emphasis added):

> Returning to the negative loop originating at decision diamond 56, when the *digital signature* of the block$_i$ *does not match* one of the signatures stored in the listing for the block, ***a change to the block$_i$ is indicated***, and the block$_i$ therefore becomes a ***candidate for back up***.

In the other interpretation, all files, or all selected files, and ALL the ***blocks*** in the file are backed up, *without determining whether any changes have been made*. In this case, examining blocks for changes does not occur and Bolt is completely irrelevant to the claimed invention.

In view of the above, Applicant submits that claim 1, and all pending claims are distinguishable over the cited references.

Notwithstanding the above, Applicant submits that the Examiner has not properly established prima facie obviousness.

As the Examiner is well aware, to properly combine references there must be some basis in the art to make the proposed combination. Applicant submits that the Examiner has not met this burden. Uemura is limited to block-based backups. As discussed throughout Uemura, a difference map is used for storing block update information. Similarly, Bolt teaches using digital signatures to identify changed blocks. Neither of these references identifies any reason to combine the respective techniques to identify changed blocks. And Levy is directed to restarting after a crash to enable transaction processing while pages are updated. Neither Levy, nor Uemura and Bolt, provide any motivation to combine the teachings of Levy with Uemura and Bolt, either alone or in combination.

In addition, while Applicant strongly submits that the proposed combination does not provide all of the features of the claimed invention, even if that were the case, the Examiner cannot use Applicant's disclosure as a road map to identify features in the prior art and impermissibly use hindsight to combine them in an attempt to arrive at Applicant's invention. In the present case, the Examiner has engaged in picking and choosing elements in Uemura, Bolt and Levy without any basis to combine them absent Applicant's disclosure.

Further, even combining the references as proposed by the Examiner does not arrive at the claimed invention. As discussed above in detail, none of Uemura, Bolt, and Levy, taken alone or in combination, teach the claimed invention, which requires, among other things:

- identifying tracks that have changed;

- identifying files for incremental backup, the identified files comprising changed and unchanged blocks saved on a track deemed changed since a last incremental backup operation; and

- incrementally backing up the identified files.

Moreover, the block-based techniques of Uemura and Bolt, are based upon detecting changes in blocks. Levy uses a fresh/stale indicators to determine which pages should be updated after a database crash. These references teach away from the claimed invention, which results in incremental backup of files containing changed and unchanged blocks on a track changed since the last incremental backup.

In addition to the above, Applicant provides some additional information regarding patentability of the claimed invention. Conventional file-based incremental backups rely on the operating system to determine if a file has been included in a previous backup. This is generally done either by examining a "last modified timestamp" on the file, or by checking a "backed up" bit stored with the file. In both cases, the backup process is relying on the operating system to maintain correct data. This approach is less than perfectly reliable, since there are circumstances (both intentional and inadvertent) under which such operating system maintained data is not accurate.

The claimed "changed track marking" in the storage system provides a highly reliable method for determining which tracks, and by inference which files, have been changed and that does not rely on the operating system to maintain this information. The result is a highly robust and independent method of changed file identification.

Further, performing a file based incremental backup (i.e., one in which all changed files are backed up in their entirety) can place a considerable load on the system I/O and computer power. The claimed invention allows the identification of changed files, as well as their backup, to take place on a secondary copy of the data that is processed by a different system dedicated to backup tasks. This approach s referred to in the specification on page 7, paragraph [0025] as a "mirrored backup".

As discussed in detail above, Uemura and Levy deal exclusively with the use of change mark ("freshness marks") to perform a block level incremental backup of part of a file. This provides substantial benefit when performing a backup of a giant database file (see, e.g., Umuera, col. 1, line 53-55) and the goal is to restore that one file to its pre-backup state.

However, this approach of significantly less benefit when backing up a more conventional file system in which there are many files (frequently tens of thousands or more), and the goal is to maintain the ability to restore any file from a piece of backup media (e.g., a single tape). In the block-based systems described by Umuera and Levy, it can be necessary to obtain the updates from an extended series of incremental backups to reconstruct a given file. This is acceptable when the alternative is a massive restore of a giant file, but less than optimal when restoring an individual file of moderate size, in which case a "full data, contiguously stored" paradigm can be more useful.

With regard to the Examiner's comments in the Response to Arguments starting on page 8 of the present application, Applicant is confused. The Examiner appears to take certain statements from Applicant's prior response and assert that since the particular order of words is different from the claims, i.e., the exact phrase is not used in the claims, the argument fails. More particularly, the following is taken from the prior response:

Claim 1, which is rejected, is set forth below with track-based incremental backup features emphasized in bold:

> A method for **incrementally backing up data** from a logically represented volume on disk media, accessible by a client through a network connection, the client comprising an enterprise database application, said method comprising:
>
> identifying **tracks** of the logically represented volume that have changed since a last **incremental backup** operation by reading fresh data indications, (i) wherein each of the fresh data indications corresponds to a **track** of the logically represented volume and (ii) wherein a given fresh data indication is indicative of whether its corresponding **track** has been changed since a last **incremental backup** operation;
>
> identifying files for **incremental backup**, the identified files comprising changed and unchanged blocks saved on a **track** deemed changed since a last **incremental backup** operation; and
>
> **incrementally backing up** the identified files from the disk media to sequential storage media through a high speed connection.

Applicant merely uses a term to refer to the entire claim with certain features highlighted. Applicant respectfully does not agree with the Examiner's characterization of Applicants arguments.

In view of the above, Applicant submits that pending claims 1-13 are in condition for allowance. Accordingly, a notice of allowance for these claims is respectfully requested.

The Examiner is respectfully invited to telephone the undersigning attorney if there are any questions regarding this Amendment or this application.

Applicant does not acquiesce to any assertion made by the Examiner that is not specifically addressed herein.

The Assistant Commissioner is hereby authorized to charge payment of any additional fees associated with this communication or credit any overpayment to Deposit Account No. 500845.

Respectfully submitted,

Dated: August 9, 2006

DALY, CROWLEY, MOFFORD & DURKEE, LLP

By: /Paul D. Durkee/

Paul D. Durkee
Reg. No. 41,003
Attorney for Applicant(s)
354A Turnpike Street, Suite 301A
Canton, MA 02021-2714
Tel.: (781) 401-9988, ext. 21
Fax: (781) 401-9966
pdd@dc-m.com

34611

Appendix:

Figure(s) 1,3 shown as both Replacement Sheet(s) and Annotated Sheet(s) Showing Changes are attached.